下载APP

# 04 | 核心链路：如何梳理符合真实业务场景的核心链路？

2021-10-26 高楼

《全链路压测实战30讲》 课程介绍 >

**讲述：高楼**

时长 10:58 大小 10.05M

你好，我是高楼。

上一讲，我给你展示了我认为一个完整的全链路压测方案应该具有的样子。在项目的选择上，我们使用的是开源微服务电商项目。这个项目采用的技术栈，是当前技术市场中流行的主流微服务技术栈，所以很可能对你也有借鉴意义。

这节课，我们还是用这个项目来讲讲，怎么梳理出符合真实业务场景的核心链路。

我们知道，在完整的企业级系统中，业务逻辑是非常复杂的。为了清晰地把业务实现到技术层面，就需要一层层地分析。

可是对一线做技术的人来说，这种做法常常会让人陷到细节里无法自拔，从而丧失全局观。所以，在做技术细节的同时，我们也很有必要再转换一下思维，从更全面的视角来观察系统。

再具体到全链路压测来看，因为我们面对的是一个完整而又复杂的线上系统，所以我们就要从架构的视角出发来分析业务链路。

不过提到架构，很多人对它并没有明确的概念，另一些人又觉得架构是个宏观的东西，无法把控。

## 从架构聊起

既然如此，我们就先来聊一聊，架构是个什么东西？

简单来说，架构就是一个系统的抽象描述，它包括系统中的组成元素以及元素与元素之间的关系。

既然是抽象，那不同的视角抽象出来的东西就不一样。像 RUP 4+1 视图，它就包括逻辑视图、开发视图、进程视图和物理视图，而场景则是用来描述他们之间的关系的。

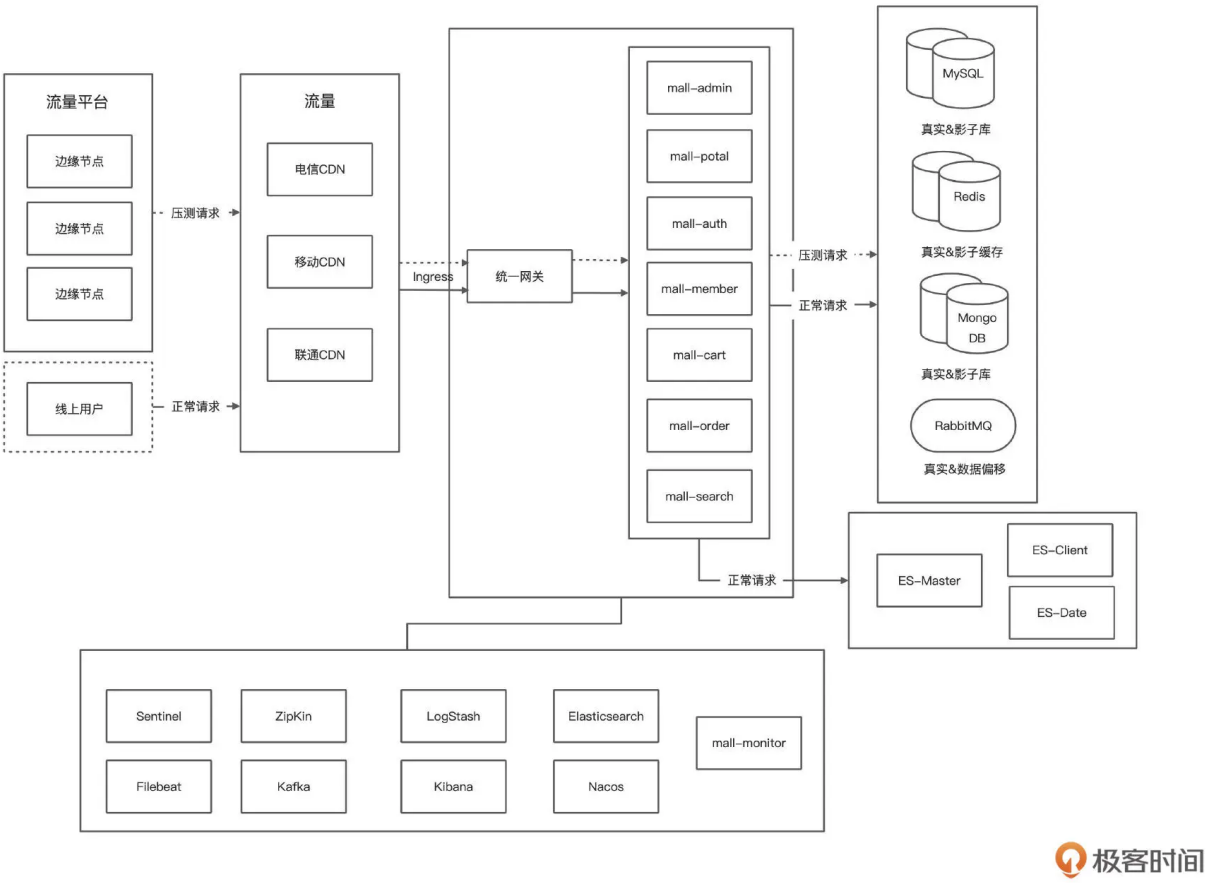
在逻辑层面，我们经常聊到的业务架构、技术架构、部署架构等，这些都是架构的不同视角。

在实现层面，我们经常聊到的单体架构、SOA 架构、微服务分布式架构、分层架构、微核架构、云架构等，这些都是从不同的技术层面来分类的。

上面说的都还只是架构的一些具体的分类。更系统地来看，技术发展到现在，已经把一个系统、完整的架构分为了四个大类，它们分别是：应用架构、技术架构、数据架构、安全架构。这是从不同的实现视角来承接业务架构的。但是这样的视角还必须落到系统中，还需要再细化到逻辑架构和物理层面，才可以最终落地。

讲了这么多架构的分类，你可能要问了，它和我们梳理业务核心链路有什么关系呢？其实讲这些是因为，链路要在架构上来画才能描述得更清楚。

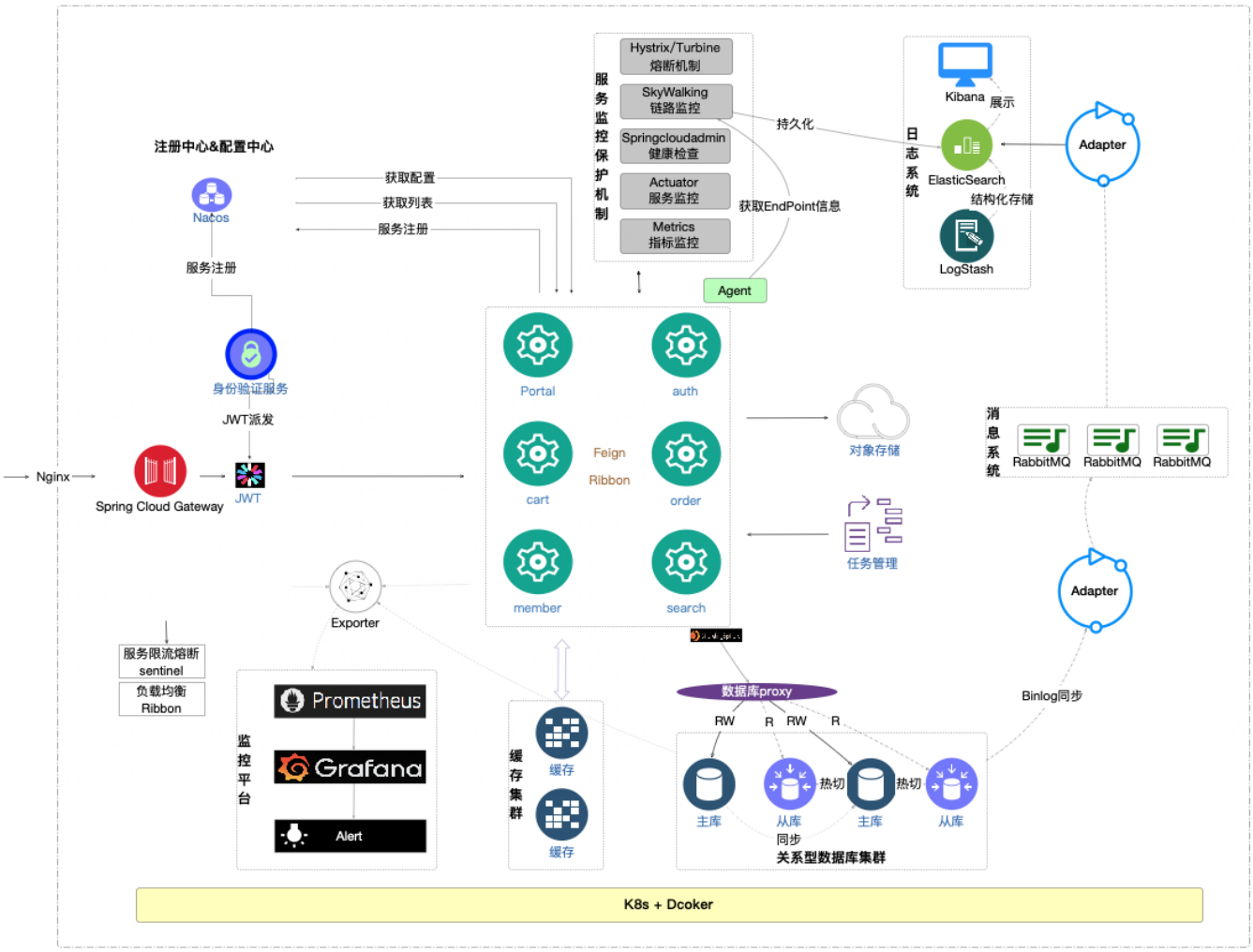
那从哪种架构开始梳理业务链路更清楚呢。先来看一下，我们这个系统应用架构的逻辑图。




像这种框框画出来的图，是不是让人看一眼大概就能知道里面有什么东西了？但这张图还没有表示出具体的部署情况。如果你觉得这个图不好看，我再放一个这样的图。

这个图是不是看起来逻辑架构清晰一点了？这张图虽然看起来和上一张图差不多，都是一些小方框，但已经开始偏向部署视角了。不过，我还没有画出服务之间的关系。

我们再换一个视角。



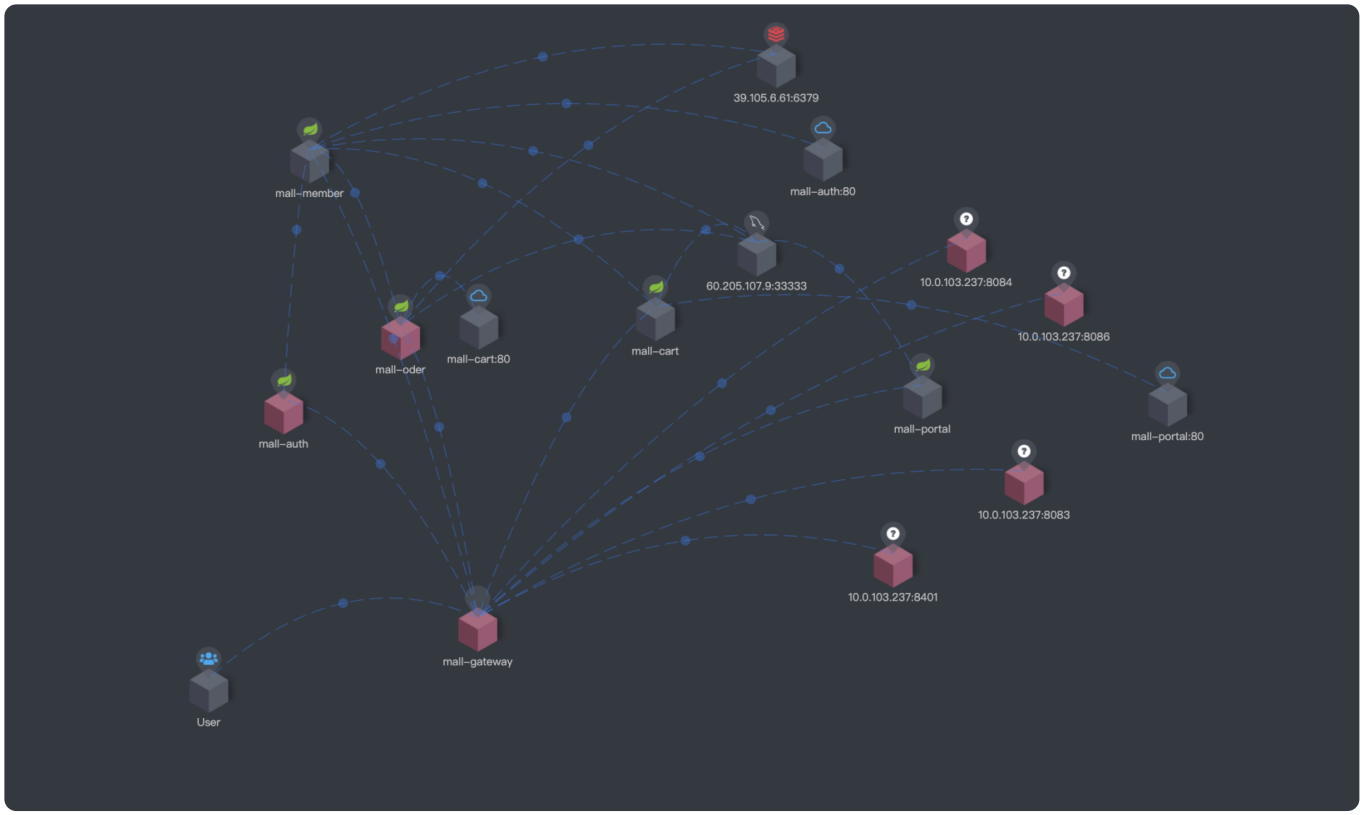

这样看起来是不是又清楚了一点？没错，这张图已经把部署视角很清晰地描绘出来了。

我这么详细地给你描述架构的不同视角，是希望你能够理解我下面的动作。我希望你能够分清楚各种不同架构的目的，因为只有理解架构才能理解链路。

## 梳理核心业务链路

但是，我们要梳理的是核心业务链路，上面的图却没有办法体现调用关系。那业务链路要怎么从这样的视角体现出来呢。

这时候我们就要祭出 APM 工具了。打个比方，对当前比较主流的微服务分布式架构来说，它的链路图是这样的。

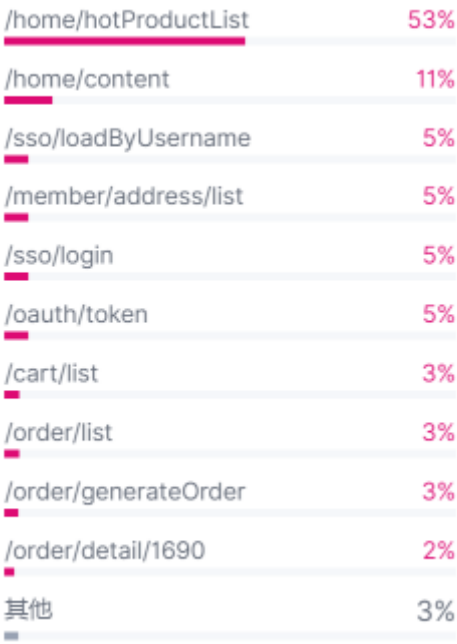



在 SOA 架构的时代，链路图都是记在脑子里的。因为服务没有那么多，倒也记得住。但是在微服务架构中，你很难记得住那么多服务。为了更有条理地梳理核心链路，我们就要借助链路监控工具了。

只有链路图还不行，要想梳理核心业务链路，还得知道什么是核心业务。核心业务的特点其实主要就两个：一、用得多；二、利润高。

我们先看符合“用得多”这个特点的业务。

先做一个业务接口访问的统计。



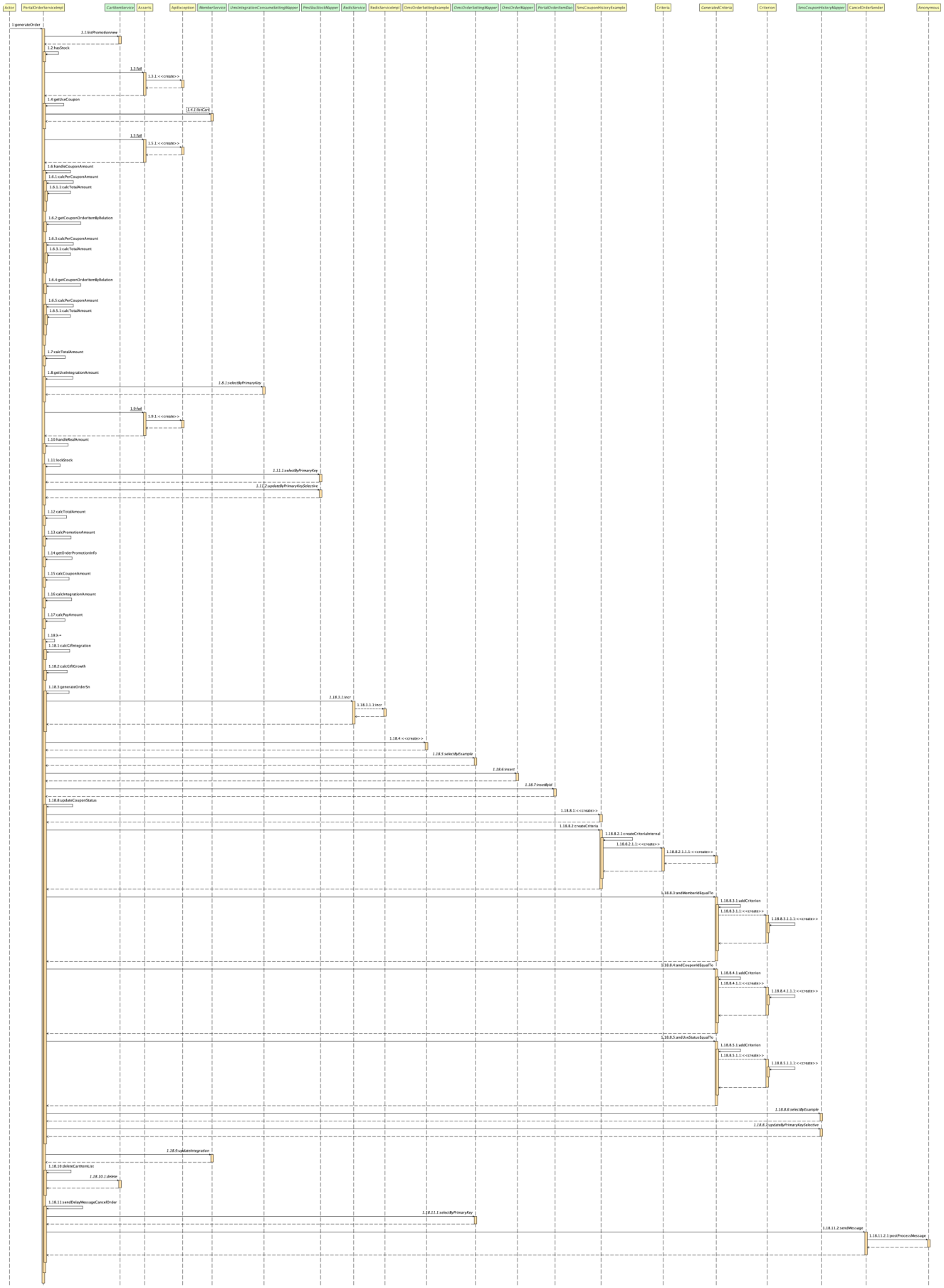

截图中展示的是统计出来的业务接口，梳理下来就是这样的业务内容。

| 序号 | 操作 | 接口 |
| --- | --- | --- |
| 1 | 打开首页 | /home/content |
| 2 | 用户登陆 | /sso/login(这个接口还会调用/oauth/token、/sso/loadByUsername) |
| 3 | 用户信息查询 | /member/address/list |
| 4 | 商品加入购物车 | /cart/add |
| 5 | 查询购物车信息 | /cart/list |
| 6 | 购物车信息确定订单 | /order/generateConfirmOrder |
| 7 | 生成订单信息 | /order/generateOrder |
| 8 | 支付前查询订单列表 | /order/list |
| 9 | 支付订单信息 | /order/paySuccess |
| 10 | 支付后查询订单详情 | /order/detail/${orderId} |

极客时间

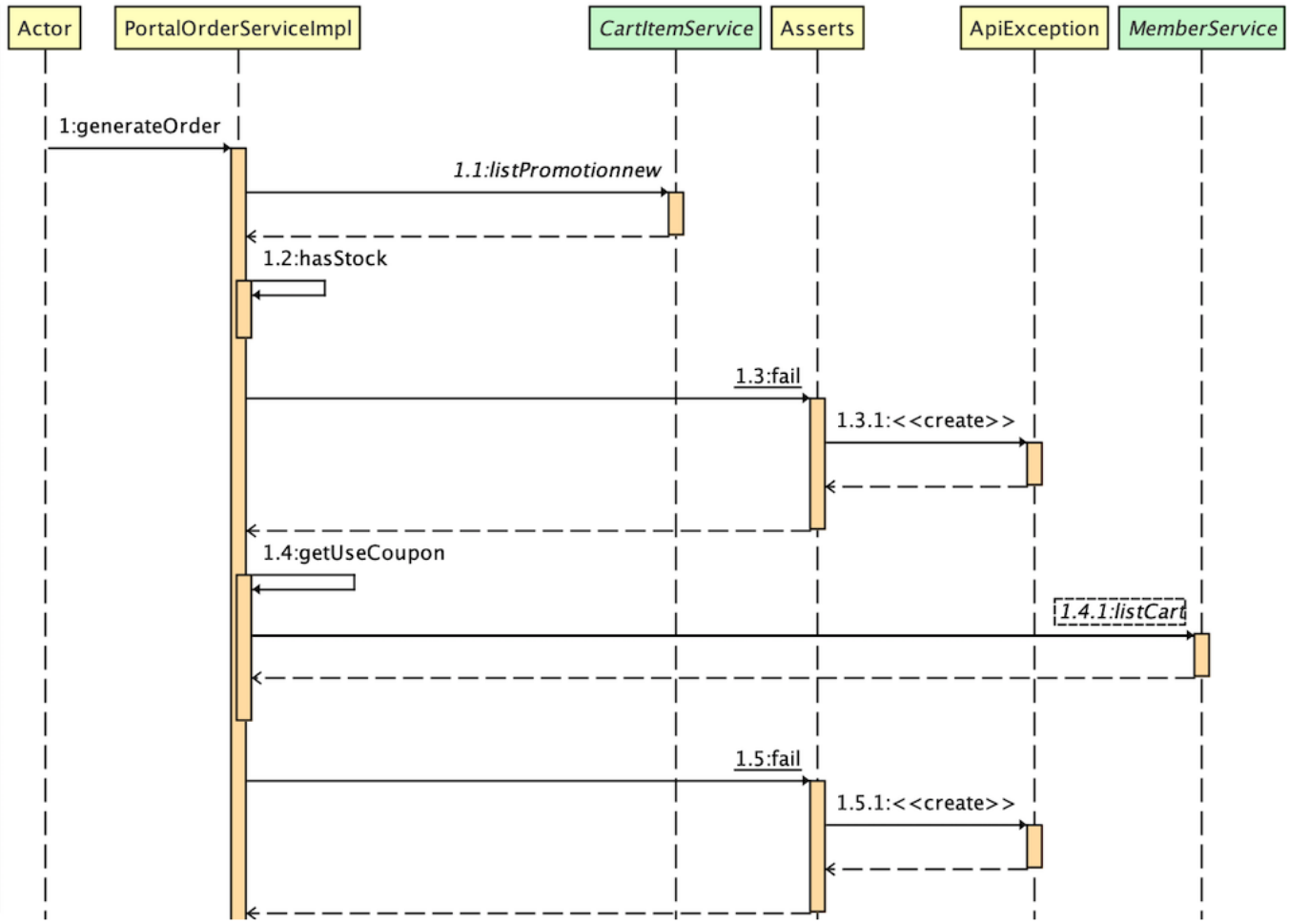
知道了是哪些业务，关键的一步就来了：你要在架构图上画出对应的链路。

我们先拿生成订单信息（在这个系统中，这个接口是最复杂的）的这个接口来说。从接口路径 /order/generateOrder 上来看，显然是直接访问的 order 接口。我们来看一下这个接口的泳道图。

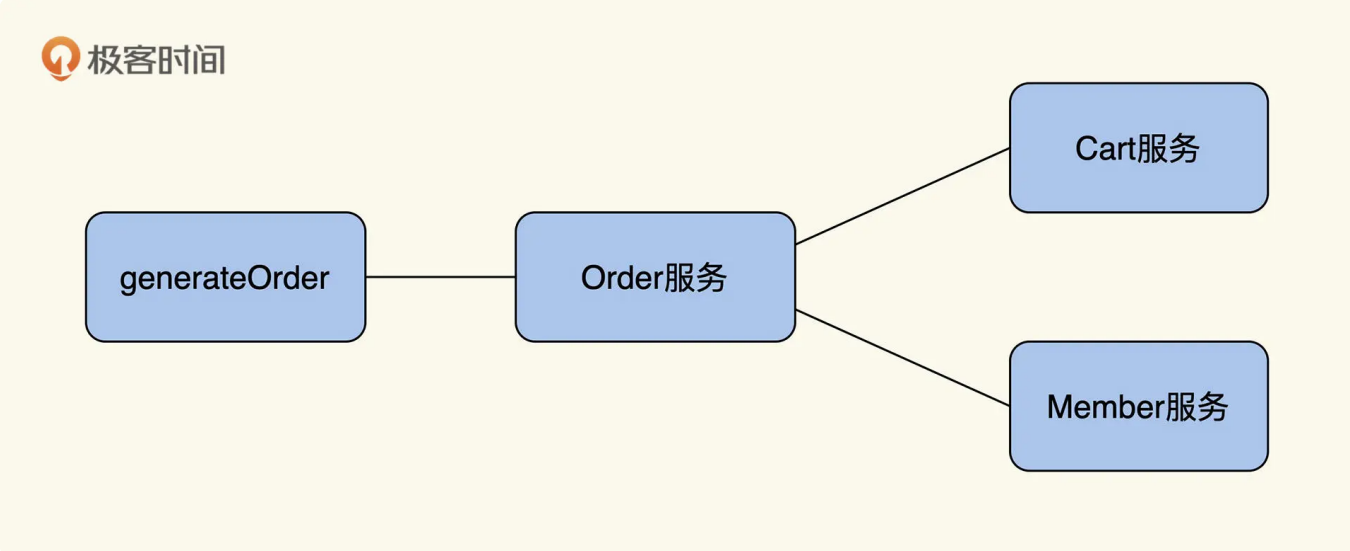



这张图是不是乍一看相当复杂？但如果仔细分析一下，你会发现它清晰明了。

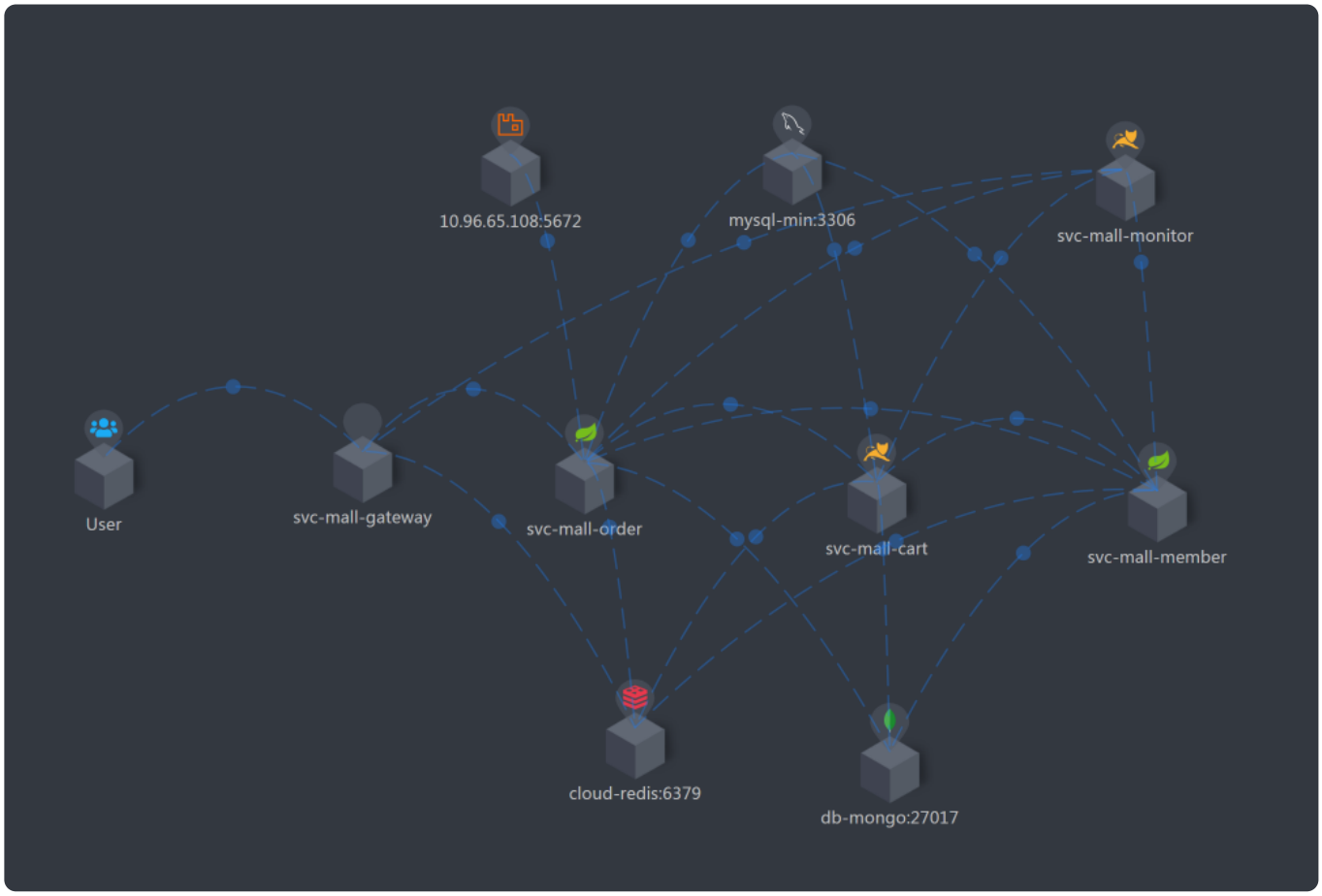
从泳道图上，你可以直观地看到和 order 直接相关的服务。因为这个图比较大，看不清楚，我们细看其中一段。



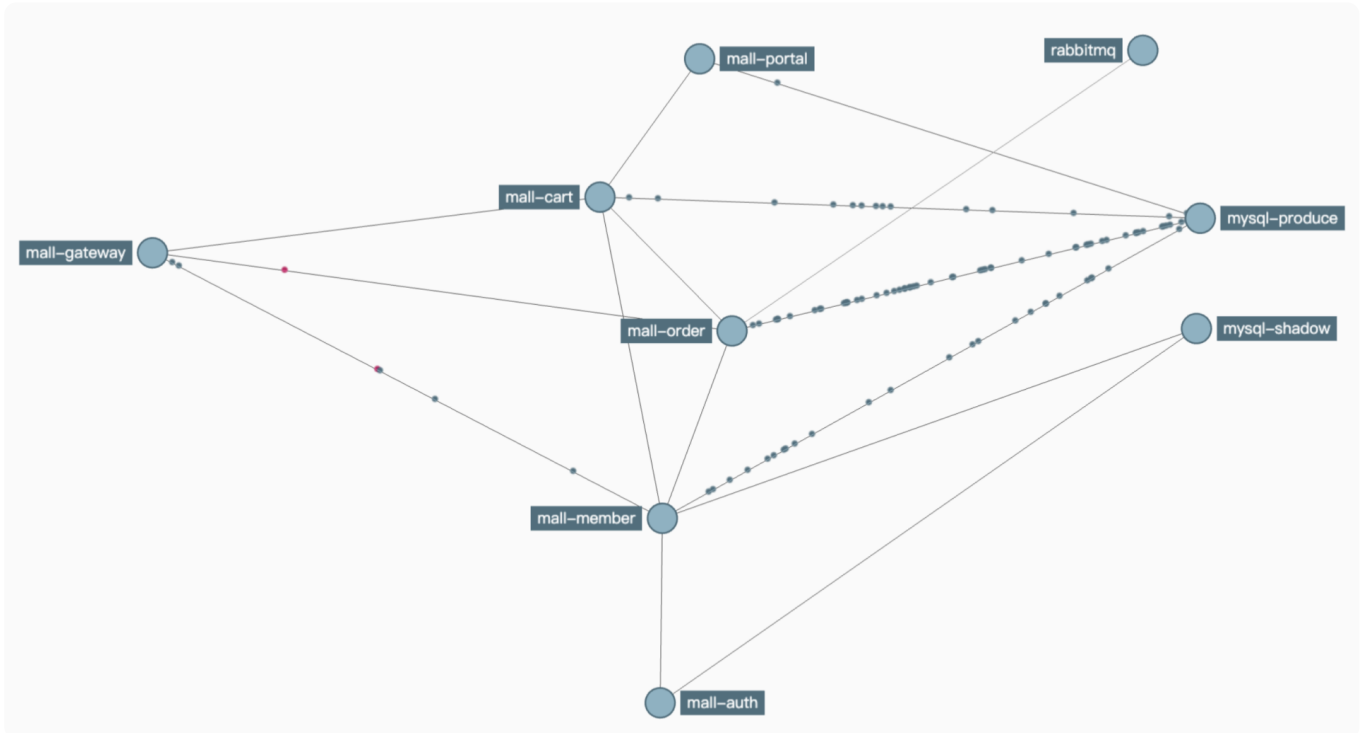

从细节图中可以看出，这个接口调用了 Cart 服务和 Member 服务。这样一来，这个链路就可以画成下面这样了。



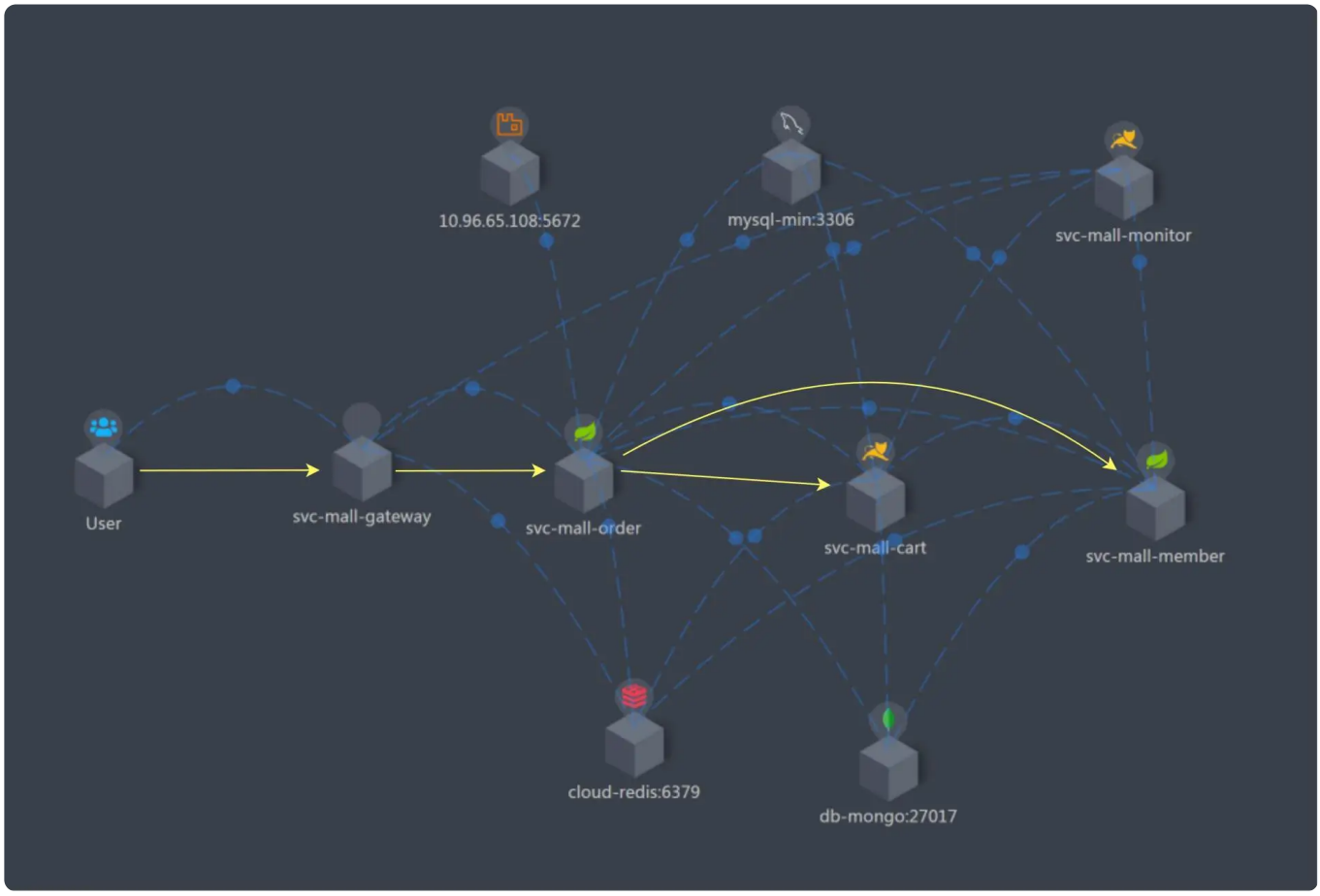
如果接着往下分析，你就能知道这个接口调用的所有链路了。但是从代码中一点点去翻链路关系显然是不理智的，借助 APM 工具 SkyWalking，我们就可以得到下面这样的图。




具体用哪种 APM 工具你可以根据自己的习惯和需要来选，只要看到相应的结构即可。比如用 Zipkin 梳理出的链路图是这样的。



在这些链路图中，你可以看到所有相关的服务，包括数据库、缓存等。不过，直晃晃地给一个图出来，还是没有办法说明链路。你还要画出箭头表示调用方向。




为了配合上面泳道图中的内容，我只画了相关的服务。这样我们就有了一个接口的链路图了。

用这样的方法把所有的接口都梳理一遍，你会得到一组完整的调用链路关系。

| 序号 | 操作 | 接口 | 调用链路 |
| --- | --- | --- | --- |
| 1 | 打开首页 | /home/content | user – gateway – portal – mysql/redis |
| 2 | 用户登陆 | /sso/login<br>(这个接口还会调用/oauth/token、/sso/loadByUsername) | user – gateway – member – mysql/redis |
| 3 | 用户信息查询 | /member/address/list | user – gateway – member – mysql/redis |
| 4 | 商品加入购物车 | /cart/add | user – gateway – cart(mysql/redis) – member – mysql/redis |
| 5 | 查询购物车信息 | /cart/list | user – gateway – cart(mysql/redis) – member – mysql/redis |
| 6 | 购物车信息确定订单 | /order/generateConfirmOrder | user – gateway – order(mysql/redis) – member – mysql/redis<br>user – gateway – order(mysql/redis) – cart – mysql/redis |
| 7 | 生成订单信息 | /order/generateOrder | user – gateway – order(mysql/redis) – member – mysql/redis |
| 8 | 支付前查询订单列表 | /order/list | user – gateway – order(mysql/redis) – member – mysql/redis |
| 9 | 支付订单信息 | /order/paySuccess | user – gateway – order(mysql/redis) – member – mysql/redis |
| 10 | 支付后查询订单详情 | /order/detail/${orderId} | user – gateway – order(mysql/redis) – member – mysql/redis |

极客时间

这样，我们就把核心业务链路都罗列出来了。

对于有第三方调用的链路，你可以直接 Mock。因为全链路压测中要屏蔽到第三方系统性能对自身系统的影响，在这一点上不用纠结。我看有些企业为了让全链路真的全起来，将第三方系统也纳入到了压测范围。从技术上说不是不可以，但是从协调上来说，就要考虑一下项目进度的风险和协调的难度了。

## 总结

好了，到这里，这节课的主要内容就讲完了。对于梳理核心业务链路来说，重点就是去看一个业务涉及到了哪些服务，这些服务又有哪些对应的技术组件。你可以遵照下面的思路：

先看架构图，看架构图是为了了解一个系统里有哪些具体的服务和组件。

再看调用链路，看调用链路是为了明确一个业务接口涉及到的具体服务和组件。

最后将所有流量大的业务接口统计出来，对应每一个业务接口梳理出调用链路。这样，我们就可以确定混合容量场景会涉及到的所有服务和组件了。

这些都是我们在后续的全链路压测过程中要重点关注的部分。

## 课后题

在结束今天的学习之前，我还给你留了两个小问题：

1. 你知道哪些确定业务接口的链路的方法？
2. 对不同的容量场景是否需要独立做业务统计？

欢迎你在留言区与我交流讨论，我们下节课再见！

分享给需要的人，Ta订阅后你可得 20 元现金奖励

生成海报并分享

赞 2 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 03 | 压测方案：你是否忽略了一个重量级文档？

## 精选留言 (1)

写留言

**BMB** 酥

2021-10-26

我们普通的项目也是这样梳理链路

展开

1 1